A Study on House Price Prediction Based on Stacking-Sorted-Weighted-Ensemble Model

Zhentao Li[1], Shiyi Xie[1*], Ying Zhang[1], Jie Hu[2]

[1] Mathematics and Computer Science Department, Guangdong Ocean University, China
[2] Modern Service Management Department, Guofang Science and Technology Technician Institute of Guangdong Province, China
1506355445@qq.com, shiyixie@126.com, 13828205545@163.com, 1793961720@qq.com

Abstract

The house price prediction problem is a typical regression problem, and most of the common house price prediction models are single prediction algorithms, which are not ideal in terms of accuracy and stability. For solving this problem, this paper proposes a house price forecasting method based on Stacking-Sorted-Weighted-Ensemble (SSWE) model. Considering the characteristics of different algorithms and giving full play to the advantages of each model, multiple individual forecasting models are fused with the Stacking model. The algorithm validation is performed using the data generated by the system of real estate management department in western Guangdong. The prediction results show that the Stacking model is superior to the single model. Compared with the Stacking regression model, the SSWE model has a 13.6% increase in the root mean square error on the training set but a 0.3% decrease on the test set, indicating that the SSWE model prevents overfitting to a some extent and increases the accuracy and stability of the model.

Keywords: Stacking, Combination forecast, Stacking-Sorted-Weighted-Ensemble

1 Introduction

As a pillar industry of national economy, real estate is inseparable from people's livelihood issues. With the intensification of urbanization, people's demand for housing rental and home purchase is increasing, and the concern for housing prices continues to become higher. Housing prices are not only related to people's living standards, but also to the smooth operation of a country's economy. Therefore, accurate prediction of house prices not only provides constructive guidance to both parties of property transactions, but also has a positive effect on the government and leaders of related departments in regulating house prices. It is important to identify an algorithm that can accurately predict house prices from the available data, which can estimate house prices based on house characteristics and predict house prices for houses with different types of characteristics.

With the rapid development of big data and machine learning, suitable algorithmic techniques, which will provide key technical support for house price prediction research, well-performance regressors such as Elastic Net (EN) [1], Random Forest (RF) [2] and Extreme Gradient Boosting [3], etc. have some applications in house price prediction studies. However, the performance improvement based on a single regressor falls into a bottleneck, such as the traditional BP neural network model has the problem of slow convergence and easy to fall into local optimum, researchers try to solve this problem by combining optimization algorithms. Magnier et al. [4] built a mixture model to predict house prices by optimizing BP neural network through genetic algorithm, the experimental results show that the convergence speed and prediction accuracy are improved, but the prediction effect is still not effectively guaranteed when the amount of data is not large enough. Therefore, some scholars have focused on the research of ensemble regressors. At present, the more mature ensemble techniques include Bagging [5-6], Boosting [7-8], and Stacking [9], which can significantly improve the performance of simple regressors such as decision trees, Random Forest is based on Bagging ensemble of decision trees [2], and XGBoost is also based on Boosting ensemble of trees [3]. The diversity and variability of the base models make the ensemble results will be more robust and accurate, which leads to a greater improvement in prediction [10]. However, the base learners of Boosting and Bagging ensemble learning are generally generated by the same learning algorithm, which cannot reflect the advantages of different algorithms. Stacking ensemble methods are applied in android malware detection [11] and emotion recognition [12], etc., which can effectively integrate different kinds of base learners, thus effectively improving the prediction accuracy. However, the selection of base learners has a large impact on the prediction results of Stacking ensemble models, and the performance of poor base learners can easily affect the combined results if the performance of base learners differs significantly. To address this problem, a mixture model is proposed and our work has three primary contributions.

- The innovation of data indicators. Most of researches about the real estate are based on the house price action of a district or the average price of a sub-district. In contrast, there are few studies of fine-grained house price forecasting that focusing on the house's own properties, such as house type, orientation, layout, etc. It is obviously that the latter can provide more valuable reference information.
- A new combination forecasting model. To solve the problem of the low prediction accuracy of single prediction model and the possibility of overfitting of ensemble learning, The SSWE is proposed, which

*Corresponding Author: Shiyi Xie; E-mail: shiyixie@126.com
DOI: 10.53106/160792642022092305022

considers the characteristics of different algorithms and makes full use of the advantages of each model.

- Real data estimation. The experiments conducted in real transaction data of western Guangdong, which is of great significance to the government's macroeconomic control of real estate and residents' home purchase decisions.

2 Related Work

2.1 Data Set

The data were collected from Departments related to real estate management, and the original data had 1,698,276 samples, each containing 70 features, and the number of screened samples was 200,356. The screening criteria are: online transfer of term houses, floor area of 44-144 square meters, house type is residential, and building structure is reinforced concrete.

2.2 Data Preparing

In the actual production environment, the data is often missing and abnormal, does not meet the needs of the model, and some of the data is invalid due to human subjective factors or objective environmental factors and other aspects, so the data must be pre-processed before modeling.

The commonly used methods for dealing with missing values fall into two categories: data filling and sample deletion, and the specific treatment depends on the analysis of the missing data. Therefore, the missing data and their missing proportions were first counted, as shown in Figure 1, and according to the statistical results, the variables with a high proportion of missing values were deleted and the null data for house orientation and house type attributes were replaced with Nan.

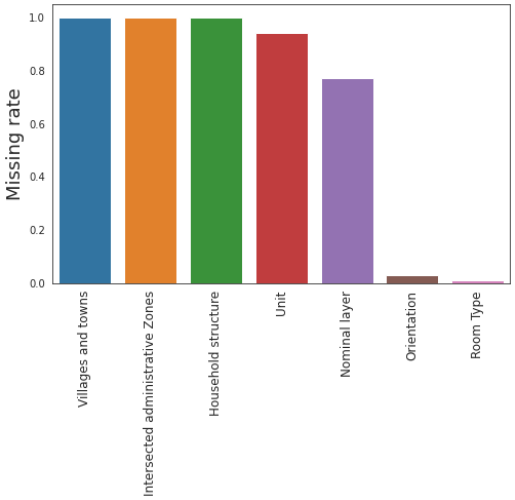

Figure 1. The statistics of missing features

The dataset used in this paper has as many as 70 variables, but some factors have very little effect on house prices, and too many features can easily lead to overfitting of the prediction results. To further filter the features, the relationship between the independent and dependent variables is visualized, and its scatter plot is shown in Figure 2.

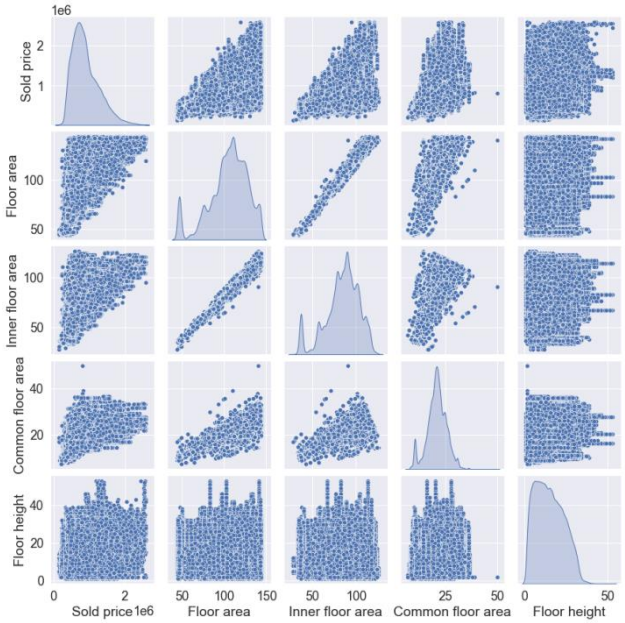

Figure 2. The Scatter plot of the relationship between the X and Y

From the above figure, it can be roughly judged that: the floor area and the Inner floor area are linearly related to the Sold price, and the common area is exponentially related to the Sold price. To further verify whether the dependent and independent variables are correlated, please see the heat map of the correlation matrix in Figure 3.

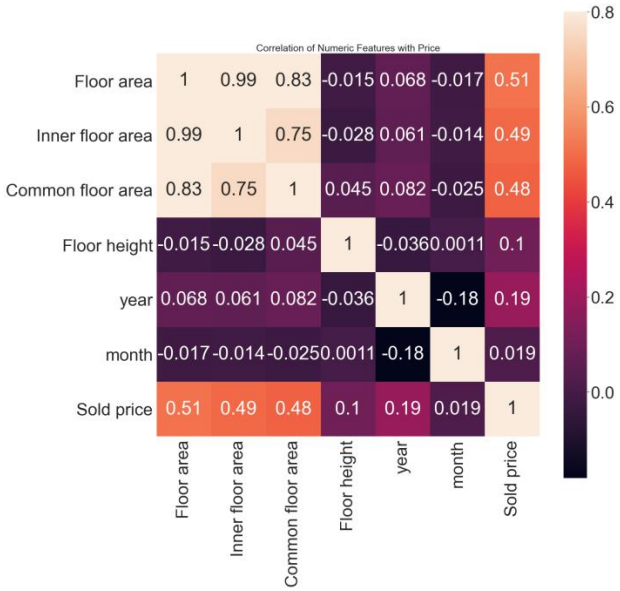

Figure 3. The related matrix heat map

The types of features in the dataset are continuous features and discrete features. One-hot coding is used for discrete features to vectorize the discrete features in the sample. n-bit state registers are used to encode N states, each corresponding to a separate register bit and only one bit is active at any given time. For continuous features, the Min-max normalization was first used for the independent variable to eliminate magnitude differences, and the Box-cox transform was used for the dependent variable to eliminate the effects of long-tailed distributions, because many models are based on the assumption that the data error terms conform to a normal

distribution, which would be violated for data with a long-tailed distribution. The Min-max normalization formula is:

$$\overline{X} = \frac{X_i - \min\limits_{1\le j\le n} X_j}{\max\limits_{1\le i\le n} X_i - \min\limits_{1\le j\le n} X_j} \quad (1)$$

Which X_i and X_j belongs to the value of the sequence and $\overline{X}$ is the transformed sequence. The Box-cox formula is:

$$y(\lambda) = \begin{cases} y^{\lambda} - 1, \lambda \neq 0 \\ \ln y, \lambda = 0 \end{cases} \quad (2)$$

Which λ is the parameter to be estimated, y is the original dependent variable and $y(\lambda)$ is the transformed response variable. The Gaussian distribution before and after the transformation of the variable is shown in Figure 4.

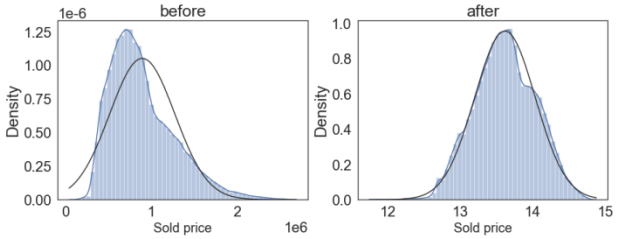

Figure 4. The Gaussian distribution before and after the transformation of Sold price (y)

3 Method

3.1 Individual Forecasting Model

3.1.1 Linear Regression

The linear regression model assumes that the regression function E(Y|X) is linear with respect to the input variables, i.e., models the relationship between one or more independent and dependent variables. The linear regression model can be defined as follows:

$$f(X) = \theta_0 + \sum_{j=1}^{n} \theta_j X_j \quad (3)$$

Where θ_j is an unknown parameter that needs to be optimized, X_j are characteristic variables. The parameters of X are solved by the least squares method. The loss function is:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(y_i - f(x_i))^2 \quad (4)$$

Calculating the first-order derivative of x for Equation (4) and Substituting Equation (3) into the previous calculation to make the result equal to 0, the analytical solution of is obtained, i.e. $\theta = (X^T X)^{-1} X^T y$. This method is simple and easy to calculate, but there are many problems. When the number of characteristic variables is large, the model is prone to drawbacks such as large variance with small deviation and poor interpretability of some variables, i.e., it arises overfitting. To solve this problem, Ridge regression and Lasso regression are proposed.

3.1.2 Ridge Regression

Ridge Regression (RR) shrinks the magnitude of the coefficients of the characteristic variables by adding a penalty factor. The loss function is expressed as:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - f(x_i)\right)^2 + \lambda \sum_{j=1}^{n} \theta_j^{\,2} \quad (5)$$

Solve for θ as $(X^T X + \lambda I)^{-1} X^T y$. Where $\lambda \sum_{j=1}^{n} \theta_j^{\,2}$ is the L_2 regularization term and λ is the penalty factor. When the penalty factor increases, the variance of the model will decrease and the bias of the model will increase. Finally, a reasonable penalty factor is found to balance the variance and bias of the model to avoid overfitting.

3.1.3 Lasso Regression

Like Ridge regression, Lasso regression also applies a penalty to the magnitude of the coefficients to solve the overfitting problem that occurs in linear regression, but the difference is that Lasso regression introduces the L1 regular term. Its loss function is as follows:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - f(x_i)\right)^2 + \lambda \sum_{j=1}^{k} |w_j| \quad (6)$$

where $\lambda \sum_{j=1}^{k} |w_j|$ is the L1 regularization term and λ is the penalty factor, which achieves the purpose of eliminating variables by making the coefficients of insignificant characteristic variables zero.

3.1.4 Decision Tree

Decision Tree is a tree-like structure that controls the generation of a tree by selecting split attributes and pruning, where each internal node represents a judgment on an attribute, each branch represents an output of a judgment result, and the leaf nodes correspond to a classification result. Common decision tree algorithms include ID3, C4.5, CART, etc.

3.1.5 Random Forest

Random Forest is a Bagging strategy for tree classification based on decision trees, in which a random selection of features is made at each node of each tree, and the most powerful feature is selected for node splitting.

3.1.6 GBDT

GBDT (Gradient Boosting Decision Tree) is an iterable algorithm consisting of multiple CART trees [13], which can be represented as follows:

$$f_t(x) = \sum_{t=1}^{T} h_t(x) \quad (7)$$

where $f_t(x)$ is t-th round model, $h_t(x)$ is the t-th decision tree. Each round of iterations produces a weak learner, each model trained on the basis of the residuals of the previous round of learners. The specific strategy is to use a forward distribution algorithm where the model at step t formed from the model at step t-1:

$$f_t(x) = f_{t-1}(x) + h_t(x) \quad \textbf{(8)}$$

The negative gradient of the loss function is used to fit the approximation of the loss in this round so that a CART regression tree can be fitted. The negative gradient of the loss function for the i-th sample of round t can be expressed as follows:

$$r_{t,i} = -\left[\frac{\delta L(y,f(x_i))}{\delta f(x_i)}\right] = y_i - f_{m-1}(x_i) \quad \textbf{(9)}$$

where $L(y, f(x_i))$ is the loss function of the true and predicted values. The t-th CART regression tree is fitted by $(x_i, r_{t,i})$ $(i = 1,2,\ldots m)$, whose corresponding leaf node region is $R_{t,j}, j = 1,2,\ldots,J$. Where J is the number of leaf nodes. The best output of the fitted leaf node $C_{t,j}$ can be defined as follows:

$$C_{t,j} = argmin \sum_{x_i \epsilon R_{t,j}} L(y_i, f_{t-1}(x_i) + c) \quad \textbf{(10)}$$

At this point the decision tree fitting function for the t-th round can be obtained:

$$h_t(x) = \sum_{j=1}^{J} C_{t,j}\, I(x \epsilon R_{t,j}) \quad \textbf{(11)}$$

Substituting into Equation (8), the final GBDT model can be obtained as follows:

$$f_t(x) = f_{t-1}(x) + \sum_{j=1}^{J} C_{t,j}\, I(x \epsilon R_{t,j}) \quad \textbf{(12)}$$

3.1.7 XGBoost

XGBoost is an efficient system implementation based on GDBT improvement. To speed up the determination of the best segmentation points, the data are pre-sorted before training and saved as a block structure so that space is traded for time to make parallelism possible. To prevent overfitting, a regularization term is introduced in the cost function, while column sampling is used by borrowing from random forests. Since XGBoost is an extension of GDBT, the objective function at step t is obtained in the same way according to Equations (7) and (8) in the previous subsection. The objective function can be defined as follows:

$$obj^t = \sum_{i=1}^{n} L(y_i, \hat{y}^{t-1} + f_t(x_i)) + \Omega(f_t) \quad \textbf{(13)}$$

where L is the loss function, y_i is the true value, $\hat{y}^{t-1}$ is the predicted value of the model at step t-1, $f_t(x_i)$ is the predicted value to be added to the new model, and is the regularization term, which can be represented as:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \quad \textbf{(14)}$$

which λ and γ are custom values, the larger the value, the simpler the tree structure, T is a leaf tree, and w_j represents the weight of leaf node j. Expanding Equation (13) with Taylor's formula and replacing the regularization term with Equation (14), Equation series (15) can be obtained:

$$obj^t = \sum_{i=1}^{n}\left[L(y_i, \hat{y}^{t-1}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \quad \textbf{(15)}$$

where g_i is the first-order derivative of the loss function and h_i is the second-order derivative of the loss function. Simplifying Equation (14) and noting that $L(y_i, \hat{y}^{t-1})$ is a constant, Equation series (16) can be obtained:

$$obj^t = \sum_{i=1}^{n}\left[\left(\sum_{i \epsilon I_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i \epsilon I_j} h_i + \lambda\right) w_j^2\right] + \gamma T \quad \textbf{(16)}$$

Taking the derivative of Equation (15) with respect to W and making it zero, the analytical solution of w_j can be obtained as:

$$w_j^* = -\frac{\sum_{i \epsilon I_j} g_i}{\sum_{i \epsilon I_j} h_i + \lambda} \quad \textbf{(17)}$$

Substituting Equation (17) into (16), the final objective function can be obtained as follows:

$$obj^t = -\frac{1}{2}\sum_{i=1}^{n} \frac{\left(\sum_{i \epsilon I_j} g_i\right)^2}{\sum_{i \epsilon I_j} h_i + \lambda} + \gamma T \quad \textbf{(18)}$$

3.1.8 LightGBM

LightGBM is another improved framework model based on GDBT, which mainly introduces two new techniques: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). GOSS can eliminate most of the samples with small gradients and use only the remaining data to calculate the information gain, because the samples with large gradients are more important for information gain, which can reduce the amount of data and ensure accurate accuracy. EFB can reduce the number of features, which can accelerate the construction of histograms.

3.2 Combination Forecasting Model

3.2.1 Stacking Model

Stacking algorithm, also known as Stacked generalization, can be regarded as a special and specific combination strategy. In order to take advantage of different algorithms, Stacking is usually heterogeneously integrated, i.e., its base learners are usually trained by different algorithms to obtain. Figure 5 shows the flowchart of the two-layer Stacking algorithm, where the first layer of learners is the primitive learner, also called the individual learner. The second layer of learners used for combining is called secondary learner or meta-learner. The basic idea is to use the initial dataset to learn a new learner by training several base learners through cross-validation and using the prediction results of these learners as input to the second layer while the original labels are still used as labels for the new dataset.

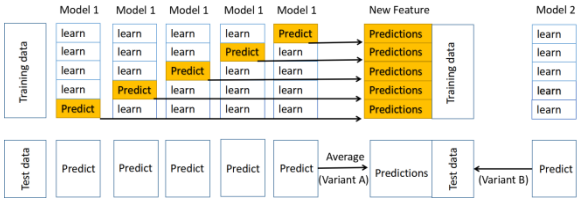

Figure 5. The flow chart of Stacking algorithm

Given a set of training samples $D = \{(x_n, y_n), n = 1, \cdots\cdots, N\}$ with the independent variable x and dependent variable y，$D_{test} = \{x_m, m = 1, \cdots\cdots, M\}$, T primary learning algorithms $\varsigma_1, \cdots, \varsigma_T$ and a secondary learning algorithm ς.

Training phase: For the training set $D = \{(x_n, y_n), n = 1, \cdots\cdots, N\}$, where x_n is the feature vector of the nth sample and y_n is the predicted value corresponding to the nth sample. Randomly divide the data into k sets of approximately equal size and mutually exclusive $D_1, D_2, \cdots, D_k$，$D_i \cap D_j = \emptyset$. $\overline{D_j} = D \backslash D_j$，$D_j$ denote the j-th fold training set and test set of k-fold cross-validation respectively. The k-fold training set is trained using the first layer learning algorithm $\varsigma_t(t = 1, \cdots, T)$ to obtain k base learners $h_t^{(j)}, j = 1, \cdots, k$. The prediction of $h_t^{(j)}$ for each sample $x_{jn}(n = 1, \cdots, N/k)$ in the j-th fold D_j of the k-fold cross-validation is denoted as $h_t^{(j)}(x_{jn})$.

Testing phase: k base learners make predictions on each sample $x_m(m = 1, \cdots, M)$ in the test set, and the prediction value is denoted as $h_t^{(j)}(x_m)$ $(j = 1, \cdots, k)$. These k results are averaged as the prediction value of algorithm ς_t on the test sample as follow

$$\overline{h_t(x_m)} = \frac{\sum_{j=1}^{k} h_t^{(j)}(x_m)}{k} \quad \textbf{(19)}$$

After the training is completed, the output data of T base learners are used as the new training set $D' = \left\{\left(y_n, h_1^{(j)}(x_{jn}), \cdots, h_T^{(j)}(x_{jn})\right), j = 1, \cdots, k \text{，} n = 1, \cdots, N/k\right\}$. The secondary learners $\varsigma(D')$ are trained by using the second layer prediction algorithm ς. The pseudo code of the training process part of Stacking ensemble learning can be represented as follows:

Input: Data set $D = \{(x_n, y_n), n = 1, \cdots\cdots, N\}$;
First layer prediction algorithm $\varsigma_1, \cdots, \varsigma_T$;
Second layer prediction algorithm ς;

Step 1: Divide the data set into k mutually exclusive subsets of equal size $D_1, D_2, \cdots, D_k$

Step 2: Generate base learners by training the first layer prediction algorithm with training data D_k

```
: for j = 1, ⋯, k do
:     for t = 1, ⋯, T do
:         h_t^(j) = ς_t(D̄_j)
:     end for
:     for x_jn ∈ D_j do
:         Calculate h_t^(j)(x_jn)
:     end for
: end for
```

Step 3: Generate a new training set $D' = \left\{\left(y_{jn}, h_1^{(j)}(x_{jn}), \cdots, h_T^{(j)}(x_{jn})\right), j = 1, \cdots, k \text{，} n = 1, \cdots, N/k\right\}$

Step 4: Each model makes predictions for each test sample x_m on the test set, and the results are averaged as: $\overline{h_t(x_m)} = \frac{\sum_{j=1}^{k} h_t^{(j)}(x_m)}{k}$

Step 5: Based on D the second layer prediction model is trained with the ς to obtain the meta-learner $h' = \varsigma(D')$

Output: $\mathbf{H(x_m) = h'(\overline{h_1(x_m)}, \overline{h_2(x_m)}, \cdots, \overline{h_T(x_m)})}$

3.2.2 Stacking-Sorted-Weighted-Ensemble Model

If a single model is used for forecasting, it may face the risk of accuracy degradation or overfitting. Dietterich pointed out that model combination may bring benefits in three aspects: statistical, computational, and representational [14]. Therefore, in this paper, the differences in the capabilities of different algorithms are used to construct SSWE models to improve the accuracy and reduce the risk of overfitting. The construction process is shown in Figure 6.

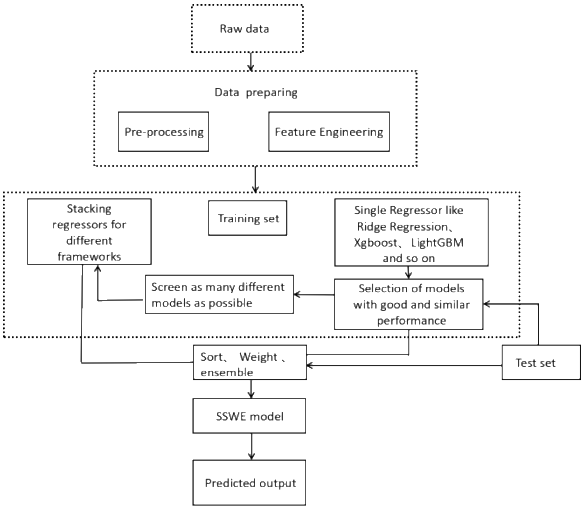

Figure 6. The flow chart of SSWE model construction

The construction process for the SSWE model is roughly divided into the following steps:

Step 1: Training base model. The training set with reconstructed samples and features is trained with Ridge Regression, Lasso Regression, Decision Tree, Random Forest, GBDT, XGBoost, and LightGBM models respectively, and the predicted values are compared with the actual values through evaluation metrics to measure the performance of each base model, The score of each model is presented in (19) below:

$$\text{Score} = (\text{val}_{\text{test}}(f_1), \text{val}_{\text{test}}(f_2), \cdots, \text{val}_{\text{test}}(f_m)) \quad \textbf{(20)}$$

where $f_m(i = 1,2, \cdots, 7)$ represents seven base prediction model respectively and val_{test} is the score of model on the test set.

Step 2: Screening model. When the model predictions are not consistent, the base prediction model with better results is selected, as shown in Equation (20):

$$f = \left[f_i \left| |min(Score) - \text{val}_{\text{test}}(f_j)| < \varepsilon \ (j \neq argmin(Score))\right.\right] \quad \textbf{(21)}$$

Step 3: Training the Stacking regressor. The Stacking framework ensemble diverse prediction algorithms that are able to take full advantage of the individual algorithms to observe data from different data spaces and structures. Therefore, the first layer of the base learner should incorporate as many different kinds of prediction algorithms as possible, in addition to selecting algorithms with excellent performance. Screening f obtained on the basis of step 2 to make it as varied as possible and the better models Ridge regression, RF

regression, GDBT regression, XGBoost regression and LightGBM regression are selected as the first layer of base learners, and Ridge regression is used as the second layer of base learners for Stacking ensemble, as shown in Figure 7.

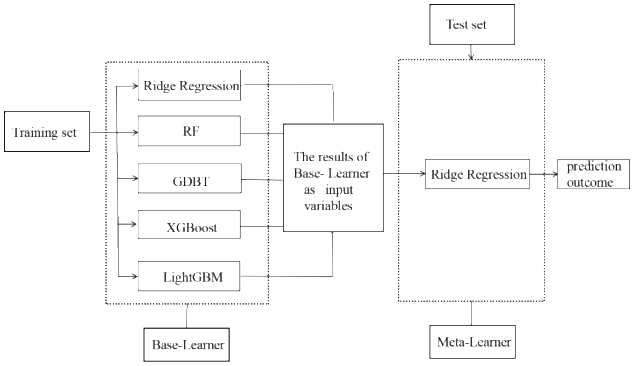

Figure 7. The workflow diagram of Stacking fusion Framework

Step 4: Combination weighting. Combining prediction models with better results according to the corresponding weights can improve the prediction accuracy of the models. The weights are used to characterize the importance of different models, considering that different models have different capabilities and contribute to the final results. The key of the combined models is the estimation of the weight coefficients, and the weights are generally set according to the minimum absolute error and variance of the objective function prediction. In this paper, we use a ranking of the performance of the base model to set the corresponding weights, it can be defined as follows：

$$F = w_i f_i \tag{22}$$

where $f_i(\mathrm{i} = 1,2,\cdots,\mathrm{n})$ represent the n well-performing underlying forecasting models, F is the combined prediction model, w_i is the weight of the i-th model, and the weights are calculated as follows

$$\begin{cases} w_i = \frac{\mathrm{n}-\mathrm{Rank}(f_i)+1}{\sum_{\mathrm{i}=1}^{\mathrm{n}}\mathrm{Rank}(f_i)}, \\ \sum_{i=1}^{n} w_i = 1, \\ w_i \geq 0. \end{cases} \tag{23}$$

where $Rank(f_i)$ is the ranking of each model's result score from highest to lowest.

3.3 Model Stability

In machine learning or statistical learning models, we often need to consider the stability of the algorithm, i.e., the robustness of the algorithm to data perturbations [15]. The generalization error of ensemble learning models is determined by the error, variance and noise, and high variance is the culprit of instability. Trade-offs between bias and variance are key in machine learning processes [16].

$$\mathrm{E}(f;D) = \mathrm{var}(\mathrm{x}) + \mathrm{bias}^2(\mathrm{x}) + \varepsilon^2 \tag{24}$$

Where $\mathrm{bias}(\mathrm{x})$ is the difference between the true value and the predicted value of the model, and the smaller the bais, the more accurate the model is. Which the $\mathrm{var}(\mathrm{x})$ reflects the error between each output of the model and the average of the model's predicted values; the smaller the variance, the more stable the model. Where $\mathrm{E}(\mathrm{f};\mathrm{D})$is the generalization error of ensemble learning model f on the unknown data set D and ε is the noise, which is the part where machine learning cannot intervene. To facilitate the illustration of the effect of bias and variance of different models, Table 1 was produced as follow.

Table 1. The effect of different bias and variance values on the model

	High bias	Small bias
High var	Not suitable for this data, need to change the model	High complexity. Overfitting.
Small var	Low complexity. Underfitting. Stable predictive values.	The generalization error is small, which is what we want to achieve.

From Table 1, when the variance of the model is large and the bias small, it will lead to overfitting (e.g., point J in Figure 8), which can be reflected in good performance in the training set but poor performance in the test set. In contrast, underfitting is inaccurate prediction for all data sets (e.g., point A in Figure 8). A good model should be accurate and stable in predicting most of the unknown data. That is, when the bias and variance are low, the generalization error of the model is low and the accuracy on unknown data is high.

The SSWE model in this paper is to prevent overfitting by balancing the relationship between variance and bias. As shown in the Figure 8, When the model complexity is high, the variance is high and the bias is low. Due to the large complexity, the model learns as much detail as possible on the training set, so the prediction is accurate, and the bias is low. However, this may lead to overfitting, which makes the model perform unstably on different data, and the model generalizes poorly, so the variance is high. Relatively, low complexity results in low variance and high bias. Due to the low complexity, the model learns more simply on the training set and cannot reach high accuracy on a certain class or set of data, so the bias will be high. But the model will predict more consistently. Although variance and bias cannot be minimized at the same time, there can be a minimum point for the generalization error consisting of both (e.g., point O in Figure 8), and we are looking for or approaching this minimum point. For models with large complexity, the variance should be reduced, and for relatively simple models, the bias should be reduced. In the SSWE screening process for the base models, all models that perform well on the test set, i.e., models with less bias (e.g., points B to I in Figure 8) are selected, based on which, models with similar performance, i.e., models with less variance (e.g., points B to G in Figure 8) are screened, and then the importance of each model is measured by ranking them, and finally they are summed, and the result can be close to the minimum point of the generalization error.

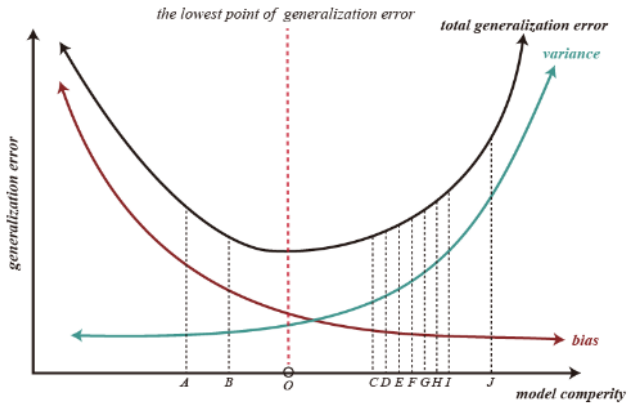

Figure 8. The plot of the relationship between model complexity and generalization error

4 Experiment

4.1 Forecasting Error Measurement

It is essential to introduce the 'forecasting error measurement' (FEM) when measuring the performance of a forecasting model. To keeping the error and house prices in same order of magnitude to describe the effect of the model, root mean square error (RMSE) is used to evaluate the prediction results of the model.

$$\mathrm{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}$$

where y_i is the true value and $\hat{y}_i$ is the predicted value.

4.2 Experimental Results

Generally speaking, the complexity of single model is low and the complexity of integrated model is high, so in this paper, experiments were done according to different model complexity to calculate the error of each model on the training and test sets, and the results are shown in Table 2. Lasso Regression (LR) , Decision Tree (DT) are all basic models with low model complexity, which obviously do not perform well on both the training set and test sets, and the bias is relatively large (e.g., point A in Figure 8), so it is not considered here as the base model for the SSWE model. ridge regression (RR), although it is also a basic model with low model complexity, performs relatively well (e.g., point B in Figure 8), and we can see that the Ridge regression model outperforms the Lasso regression model, and we can consider it as the base model for the SSWE model, and the result has the added benefit that it can be inferred from this that the characteristic variables chosen in this paper all have a larger role. And the reader can also test other single models to do an extended study. Among the underlying prediction models, the LightGBM regression model performs the best. Although the LightGBM regression prediction model does not achieve the lowest RMSE in the training set, it have the smallest RMSE = 0.1490973 in the test set. RF have the smallest RMSE in the training set, but is less effective in the test set and overfitting occurred. The Ridge regression model outperforms the Lasso regression model, and suggesting that all the characteristic variables have a greater effect. Based on the performance of the basic models, the better models Ridge regression, RF regression, GDBT regression, XGBoost regression and LightGBM regression were selected as the first layer base learner and Ridge regression as the second layer base learner for Stacking integration, compared to the LightGBM regression model, there was a 72.9% reduction in the training set and a 1.3% reduction in the test set. It indicating that the Stacking model is superior to the single prediction model. The SSWE model was constructed by selecting Stacking ensemble regression model, RF regression model, GDBT regression model, XGBoost regression, and LightGBM regression, and the RMSE became larger in the training set but reduced by 0.3% in the test set compared to the Stacking ensemble regression model. Therefore, the SSWE model is superior to the Stacking ensemble model and better than the single model such as Ridge regression in terms of various prediction evaluation metrics in the training and test sets.

Table 2. RMSE of values of different models

	RMSE	
Model	Training set	Test set
Lasso Regression	0.4257673	0.4104566
Ridge Regression	0.1385489	0.1765388
Decision Tree	0.1205079	0.1801412
RF	0.1006223	0.1562953
GDBT	0.1070908	0.1500351
XGBoost	0.1095310	0.1518729
LightGBM	0.1013876	0.1490973
Stacking	0.0274669	0.1470694
SSWE	0.0312009	0.1465776

5 Conclusion

In AI house price prediction research, most scholars predict real estate transaction prices from the macro level, which can only predict the trend direction of average house price and has little significance to the government's macro control and residents' choices. This paper adopts data indicator innovation to predict house prices based on housing portrait indicators from the micro level to achieve accurate prediction of one house at one price, and the data set is collected from the actual production process, which is of strong practical significance and provides relevant departments with ideas for processing this type of data set. In addition, this study also proposes a SSWE model, which combines the Stacking ensemble model and the single base model in a weighted combination, the smaller the RMSE of the model on the test set, the higher the ranking and the greater the weight it takes, and the five models with better prediction effect is obtained: Ridge regression, LightGBM regression, RF regression, GBDT regression, XGB regression, and Stacking ensemble model. Combining these models using the method just mentioned, the final SSWE model is obtained. The advantage of the SSWE model is that it takes into account the diversity and variability of the underlying model to make the prediction results more robust and accurate, which not only improves the prediction ability of house prices to some extent, but also effectively avoids the occurrence of overfitting when the data has noise or more features, and greatly improves the stability of the model.

In future work, we can try to incorporate or explore some more advanced techniques or features to improve the prediction, but the model trained with state-of-the-art algorithms means that there may be a greater model complexity, and whether the model complexity corresponding to the algorithm seems too high can be judged by doing cross-validation and then finding the variance, which of course means higher computational and time costs. The lowest point of generalization error is actually a theoretical value, and how to approach this point may be the direction of future SSWE model improvement.

References

[1] H. Zou, T. Hastie, Regularization and Variable Selection via the Elastic Net, *Journal of the Royal Statistical Society*, Vol. 67, No. 2, pp. 301-320, April, 2005.

[2] L. Breiman, Random Forests, *Machine Learning*, Vol. 45, No. 1, pp. 5-32, October, 2001.

[3] T. Q. Chen, C. Guestrin, Xgboost: A Scalable Tree Boosting System, *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, California USA, 2016, pp. 785-794.

[4] L. Magnier, F. Haghighat, Multiobjective optimization of building design using TRNSYS simulations, genetic algorithm, and Artificial Neural Network, *Building and Environment*, Vol. 45, No. 3, pp. 739-746, March, 2010.

[5] L. Breiman, Bagging Predictors, *Machine Learning*, Vol. 24, No. 2, pp. 123-140, August, 1996.

[6] F. L. John, J. P. J. John, Randomly Selected Heterogenic Bagging with Cognitive Entity Metrics for Prediction of Heterogeneous Defects, *International Journal of Performability Engineering*, Vol. 17, No. 9, pp. 796-803, September, 2021.

[7] M. Schonlau, Boosted Regression (Boosting): An Introductory Tutorial and a Stata Plugin, *The Stata Journal*, Vol 5, No. 3, pp. 330-354, September, 2005.

[8] M. A. Beghoura, Software Engineering Teamwork Data Understanding using an Embedded Feature Selection, *International Journal of Performability Engineering*, Vol. 17, No. 5, pp. 464-472, May, 2021.

[9] D. H. Wolpert, Stacked Generalization, *Neural Networks,* Vol. 5, No. 2, pp. 241-259, 1992.

[10] G. Brown, J. Wyatt, R. Harris, Y. Xin, Diversity Creation methods: a Survey and Categorization, *Information Fusion*, Vol. 6, No. 1, pp. 5-20, March, 2005.

[11] H. J. Zhu, Y. Li, R. D. Li, J. Q. Li, Z. H. You, H. B. Song, SEDMDroid:An Enhanced Stacking Ensemble Framework for Android Malware Detection, *IEEE Transactions on Network Science and Engineering*, Vol. 8, No. 2, pp. 984-994, April-June, 2021.

[12] Q. J. Kang, Q. Gao, Y. Song, Z. K. Tian, Y. Yang, Z. M. Mao, E. Z. Dong, Emotion Recognition from EEG Signals of Hearing-Impaired People Using Stacking Ensemble Learning Framework Based on a Novel Brain Network, *IEEE Sensors Journal*, Vol. 21, No. 20, pp. 23245-23255, October, 2021.

[13] J. H. Friedman, Greedy Function Approximation: A Gradient boosting Machine, *Annals of Statistics*, Vol. 29, No. 5, pp. 1189-1232, October, 2001.

[14] T. G. Dietterich, Ensemble Methods in Machine Learning, *International Workshop on Multiple Classifier Systems,* Cagliari, Italy, 2000, pp. 1-15.

[15] Y. T. Hu, C. S. Gu, Z. Z. Meng, C. F. Shao, Improve the Model Stability of Dam's Displacement Prediction Using a Numerical-Statistical Combined Model, *Institute of Electrical and Electronics Engineers*, Vol. 8, pp. 147482-147493, August, 2020.

[16] P. L. Espinheria, L. C. M. Silva, F. Cribari-Neto, Bias and variance residuals for machine learning nonlinear simplex regressions, *Expert Systems with Applications*, Vol. 185, pp. 115656, December, 2021.

Biographies

Zhentao Li is studying for his master's degree in the School of Mathematics and Computer Science at Guangdong Ocean University, China. His research interests include Data processing, machine learning and image processing.

Shiyi Xie graduated from Tsinghua University, for the degree of Master. He is currently a professor in computers at Guangdong Ocean University, China. His research interests include Digital Ocean, Internet of Things and Big Data Technologies.

Ying Zhang graduated from Jilin University, for the degree of Ph. D and worked as a postdoctoral researcher. She is currently an associate professor in computers at Guangdong Ocean University, China. Her research interests include data mining and geological applications.

Jie Hu graduated from Guangdong Ocean University, for the degree of Master. She has been a faculty with the School of Guofang Science and Technology Technician Institute of Guangdong Province, China. Her research interests include data mining and Big Data Technologies.